Fraunhofer FOKUS

FAME
Future Applications
and Media

VERSANT

Benchmarking the Versant Object Database and MySQL with the FOKUS Open IPTV Ecosystem

White Paper

Publisher:
Fraunhofer Institute for Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
Phone +49 30 3463 7000
info@fokus.fraunhofer.de
www.fokus.fraunhofer.de

Versant Corporation
255 Shoreline Drive, Suite 450,
Redwood City, CA 94065
Phone +1 650-232-2400
info@versant.com
www.versant.com

Design and Layout:
Fuel Design
Kirsten Felbert

Picturesource:
iStock

Contact:
Fraunhofer Institute for Open Communication Systems FOKUS
Competence Center Future Applications and Media
Dr. Stefan Arbanowski
Phone +49 30 3463 7197
stefan.arbanowski@fokus.fraunhofer.de

Versant Corporation
Dirk Bartels
Phone +1 650-232-2432
dbartels@versant.com

Benchmarking the Versant Object Database and MySQL with the FOKUS Open IPTV Ecosystem

White Paper

Abstract

Databases for managing and controlling IPTV content are indispensable for an IPTV platform. In this application domain, the database must provide fast access to rather complex data, easy integration into an ever evolving application code, and support massive concurrent access from potentially millions of consumers. Choosing a database for such a system and the reasons this choice serve as the subject of this paper.

We have implemented Fraunhofer FOKUS Open IPTV Ecosystem with two database systems: The Versant Object Database supports a native object-oriented data model which allows managing application objects directly in the database. Oracle's MySQL, by contrast, a popular open source relational database, requires an object-to-relational mapping layer to manage the application objects in the relational data model.

Analyzing the results of a number of IPTV related benchmark use cases should help the reader to evaluate the pros and cons of each of these two database systems in such an environment.

The FOKUS IPTV Ecosystem

What is IPTV?

There are many definitions of IPTV. Most commonly, it stands for Internet Protocol Television, where television service is delivered using Internet Protocol over a broadband network. The evolving IPTV service has many advantages and potential over traditional broadcast TV service, because it can provide a more personalized and interactive environment and experience. IPTV can be delivered using a variety of networks, including Managed Networks (an end-to-end network managed by an operator) and the Open Internet. With traditional TV services, consumers are able to watch scheduled programs where the only interaction possible is the ability to change channels. IPTV not only allows users to interact with scheduled programs, such as voting with their remote control. It also provides Content on Demand, where users select content items they want to watch and when. The Content Guide can be customized by the user, and can also integrate scheduled programs and

on-demand items. All of these features are provided without using any additional or dedicated devices.

What is the Open IPTV Standard?

There is a growing standardization effort for the use of the 3GPP IP Multimedia Subsystem (IMS) as an architecture to support IPTV services in carriers networks. Both ITU-T and ETSI are working on so-called "IMS-based IPTV" standards. Carriers will be able to offer both voice and IPTV services over the same core infrastructure and the implementation of services combining conventional TV services with telephony features (e.g., caller ID on the TV screen) will become straightforward.

The Open IPTV Forum is creating specifications for an open end-to-end solution for IPTV for both the Managed Network (e.g., IMS) and Open Internet architectures. This brings new opportunities to players in the IPTV value chain for supplying a variety of IPTV and Internet multimedia services to retail-based consumer equipment in the home.

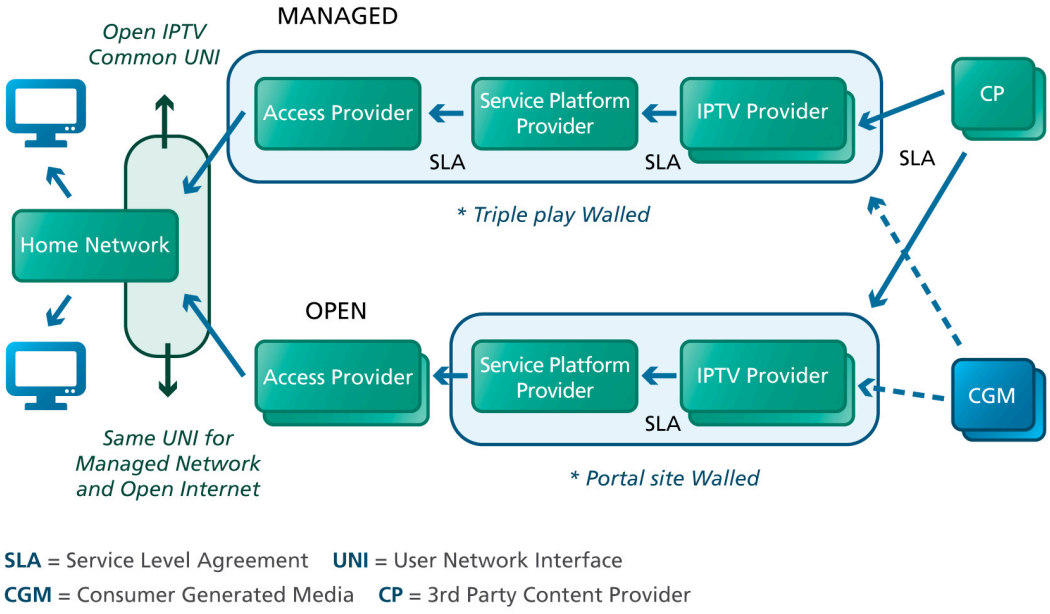

Figure 1: Open IPTV Forum scope

What is the FOKUS Open IPTV Ecosystem?

System Overview

The FOKUS Open IPTV Ecosystem (see figure 2) consists of IPTV Clients acting as IPTV Terminal Functions (ITF) and IMS User Agents (UA) that align with ETSI TISPAN Release 2, Open IPTV Forum Release 1 and DVB specifications. The software is available for Microsoft Windows platforms (Vista, XP, XP embedded), Linux and as Rich Internet Applications using Microsoft Silverlight and JavaFX. The FOKUS Open IPTV Ecosystem incorporates Next Generation Networks (NGN) signaling towards telco infrastructures and supports all common NGN functionalities, including VoIP telephony, Instant Messaging, Presence Services and TISPAN/OITF IPTV signaling for channel and content selection. Media Control and delivery is handled, via the RTSP protocol using RTP streaming.

The IPTV Service Control (IPTVSC) is the main part of the FOKUS Open IPTV Ecosystem. It is an HTTP/SIP

application server (Java servlet running on Sailfin Glassfish app server 1.5) managing all IPTV related data. It uses several repositories, e.g., to store IPTV content, sessions, states, and log entries. The Versant Object Database (V/OD) and MySQL database systems are integrated into this component via the Java Data Objects (JDO) API 2.0, an industry standard persistence specification for the Java developer.

Following is a short description of these main components:

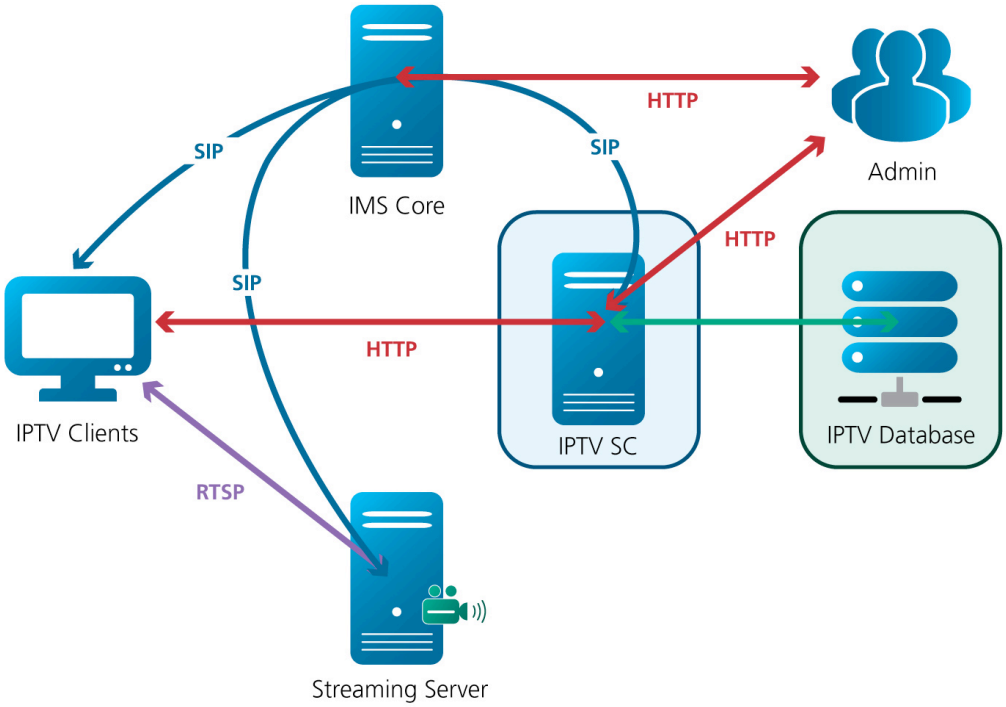

Figure 2: FOKUS Open IPTV Ecosystem

IPTV Session Control

The IPTV SC is separated into 4 components. Each component uses collections of data called repositories. A repository is a logical database for certain domain objects (e.g., IPTV User Repository for IPTV User objects). All repositories are managed in the same physical database.

IPTV SC Components

- IPTV Service Discovery

  The IPTV Service Discovery is an SIP interface to offer Service Provider and Service Discovery Information. It is the entry point for an IPTV Client after registration at the IMS Core.

- IPTV Service Selection

  The IPTV Service Discovery is an HTTP interface which offers Service Provider and Service Discovery Information, as well as Broadband Content Guide (BCG) Information, provided in a so-called TV-Anytime XML format.

- IPTV Service Control

  The IPTV Service Control is an SIP interface to manage and log IPTV sessions.

- IPTV Content Management

The IPTV Content Management is an HTTP interface providing a HTML admin console for managing and controlling the IPTV SC.

IPTV SC Repositories

- IPTV User Repository

The IPTV User Repository holds the IPTV users and their current state.

- IPTV Session Log Entry Repository
  The IPTV Session Log Entry Repository stores all initiated sessions.
- IPTV Media Container Repository
  The IPTV Media Container Repository contains all media items with basic information.
- IPTV Profile Repository

The IPTV Profile Repository contains IPTV profiles providing personalized content.

- IPTV Service Repository
  The IPTV Service Repository contains IPTV services to provide different access to content.
- IPTV Media Collection Repository
  The Media Collection Repository holds the BCG data that will be used to generate the TV-Anytime XML. This repository will be automatically filled by services using external Electronic Program Guide (EPG) servers.

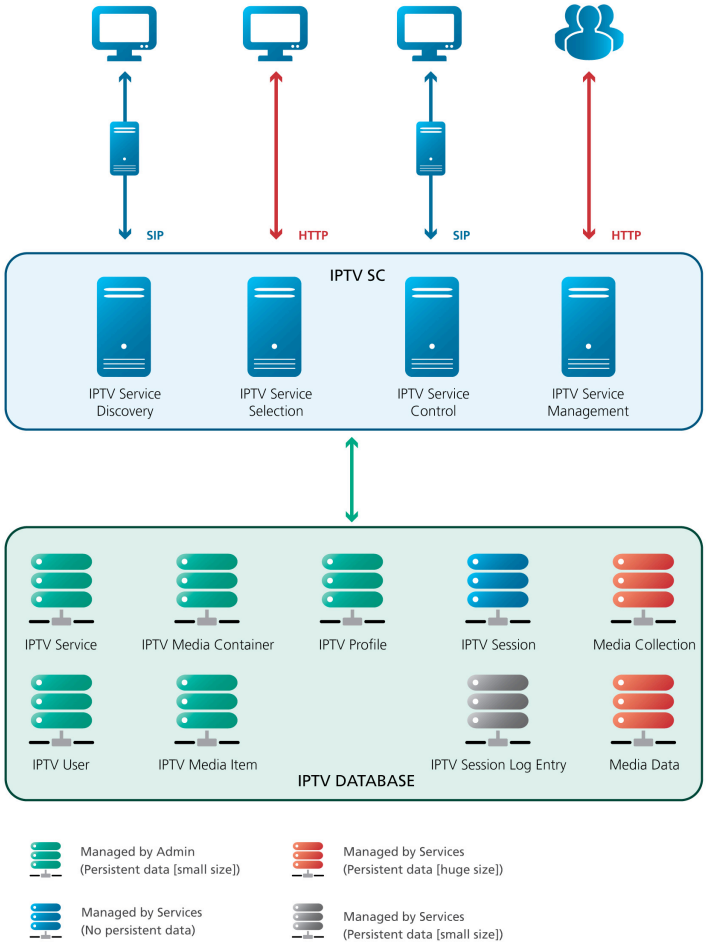

Figure 3: IPTV SC + IPTV Database

An IPTV Database Benchmark

Main Objectives

The benchmark provides quantitative results to help compare and analyze performance, complexity, scalability, concurrency capabilities and total costs of ownership of the tested ecosystem using either V/OD or the MySQL database system. Fraunhofer FOKUS also evaluated the JDO standard programming interface and compared it to a native MySQL implementation, e.g., in regard to code usability, integration, and extensibility. Lastly, the compatibility of the JDO implementation of V/OD and MySQL using the DataNucleus object-to-relational mapping tool was evaluated.

The IPTV Benchmark Use Cases

Use Case 1: IPTV SC

The IPTV SC must handle concurrent IPTV sessions depending on the number of subscribers. The first benchmark use case creates and terminates IPTV sessions for a variable number of concurrent users (1, 100, 10k). The main requirement is to handle the demanding number of database read/write operations and database queries with small-sized data objects simultaneously (e.g., highly concurrent transactions). The benchmark records the time for a single SIP session initiation/termination request, the single request execution

time on the server, as well as the elapsed time for each test run. Furthermore, this use case analyses the concurrency capabilities of the database.

SIP requests are sent directly to the IPTV SC and not routed through the IMS Core to avoid measuring any unrelated overhead. The Open Source test tool SIPp was used to generate the SIP requests.

Figure 4 shows the network and database message flow for a single test run.

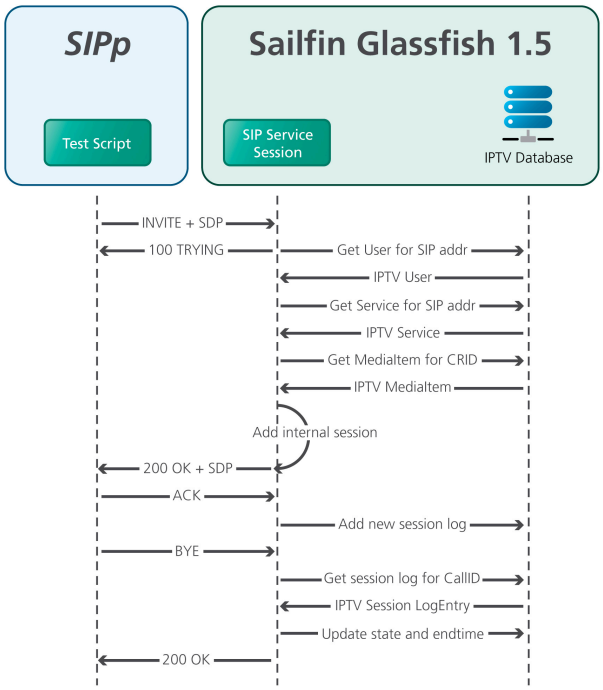

Figure 4: Benchmark use case 1 message flow

The first SIP INVITE Request, including the Session Description Protocol (SDP), creates a logical session on the Sailfin server, fetches related data for the session from the database and sends the result response. The first request only contains read operations for the database which contains small sized data objects like requesting user, requested service, and requested media. After the result response the client answers with a SIP ACK Request to confirm the response. This request creates a session log entry into the database, thus it has just a single write operation. After this, the session is up. To terminate the session, the client sends a SIP BYE Request. This request fetches the session log entry and updates its state. Thus, the request only has one read and write operation for the database. The server confirms the termination with a response. After this, the session is down and one single test run is completed.

Use Case 2: Broadband Content Guide

The Besides handling small-sized data objects simultaneously, the IPTV SC also manages larger sized BCG data. In the real world, the IPTV SC updates the BCG data at least once a day. The update process collects EPG data from external servers, converts the raw EPG data into Media Data objects, and subsequently stores them into the local database.

This use case measures the time for a complete BCG update process for 10, 30, and 50 channels. Each channel provides EPG information for 7 days, including about 20 program entries per day (with the formal:

data entries count = channels count* 7days * 20 programs/day * 50 metadata entries/program, we get about 70,000 data entries for 10 channels, 210,000 for 30, and 350,000 for 50). The test case writes large sized data objects at a high frequency into the database in a single transaction. The test case helps to analyze how well the database handles data complexity and scalability with larger data sets. Additionally, the benchmark records the time for reading the refreshed BCG data.

The message flow for this use case is illustrated in figure 5.

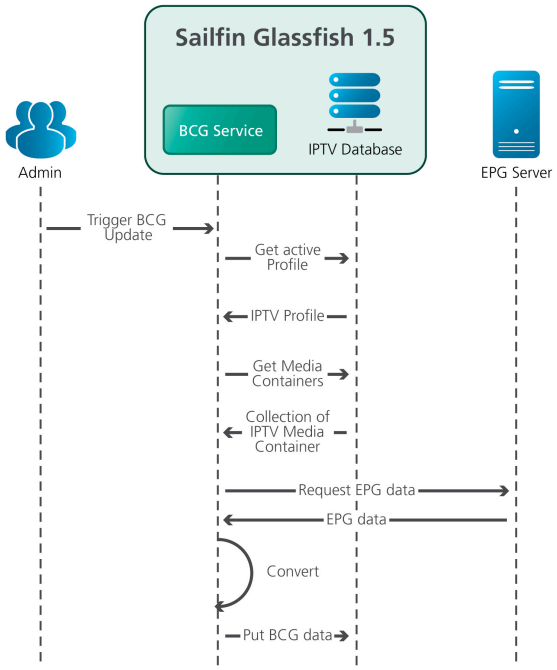

Figure 5: Benchmark use case 2 message flow

After triggering the refresh process, the server fetches some related information from the database like profile and media container. Afterwards the server collects raw EPG data from external servers and converts it to persistence-capable BCG data. Finally, the server writes all converted BCG data into the database in a single transaction.

Benchmark Environment

- General Operating Environment

  The Sailfin Glassfish application server is running on a Windows Server 2008 Standard Edition with 4 GB RAM in a Virtual Hyper-V instance. The host machine is a Windows Server 2008 64 Bit Enterprise running on an IBM x3650 19inch, including 2x Quad-Core Intel Xeon Processor E5420 and 16GB RAM. The Sailfin server is configured with default settings.

- MySQL Database System

  A MySQL Server 5.0 configured with default settings is running on a local server machine (same sever where the Sailfin App Server is running). The access to the MySQL database is implemented via the DataNucleus JDO object-to-relational mapping tool.

- Versant Object Database System V/DO

  A V/OD Server 8.0.1 with default settings is running on a local server machine (same server where the Sailfin App Server is running). Access to V/OD is implemented via the Versant JDO application programming interface.

Implementation Remarks

Application and database design

The IPTV SC components are running as SIP/HTTP Java servlets on the Sailfin Glassfish application server. The implementation is based on the Fraunhofer FOKUS Composite Java Application Framework (COJAL), a module-based framework providing global configuration settings, module/unity/event manager, dependency injections, and common logging.

Each repository is registered as a COJAL unity (only one instance per repository) that is used by components to get access to the underlying database via the Persistence Manager. The Persistence Manager and the Persistence Manager Factory will be efficiently pooled inside each request thread. The Persistence Manager implementations (Versant, respectively DataNucleus) keep the repositories synchronized.

The implementation allows changing the database connection on-the-fly via the IPTV SC admin console to easily allow comparing both database systems. A screenshot of the admin console is displayed in figure 6.

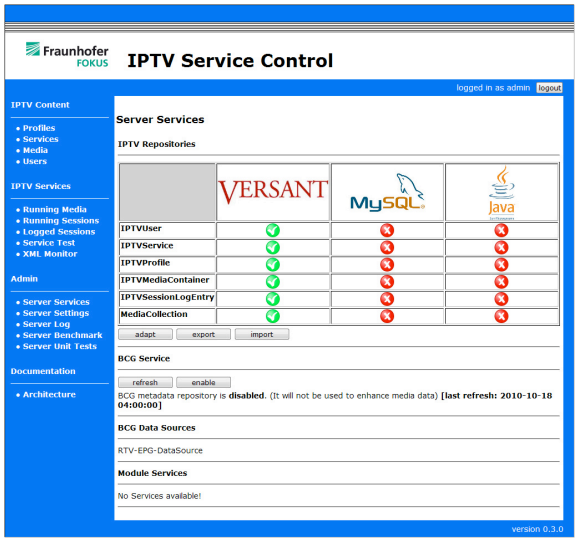

Figure 6: IPTV SC admin console

An abstraction of the IPTV SC application model is shown in figure 7.

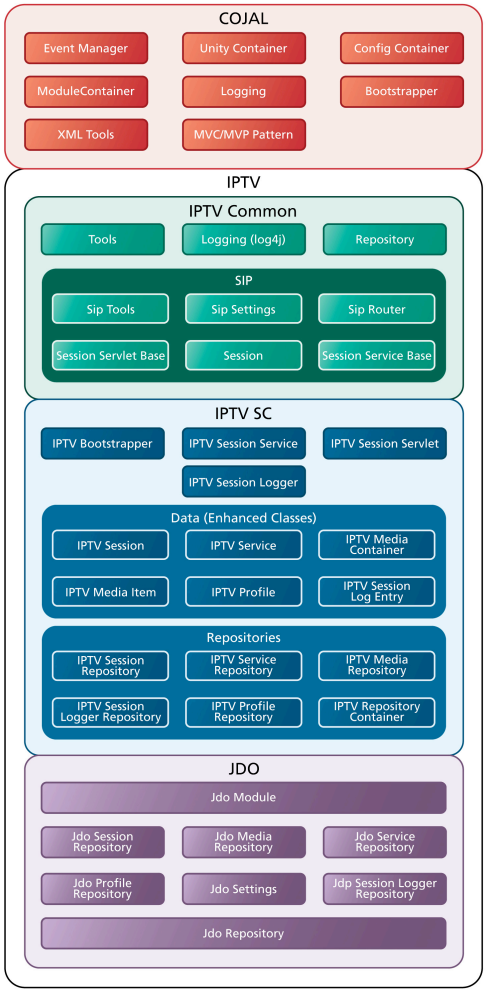

Figure 7: IPTV SC application model

The database schema for the IPTV data objects and the BCG data is shown in figure 8 and 9.

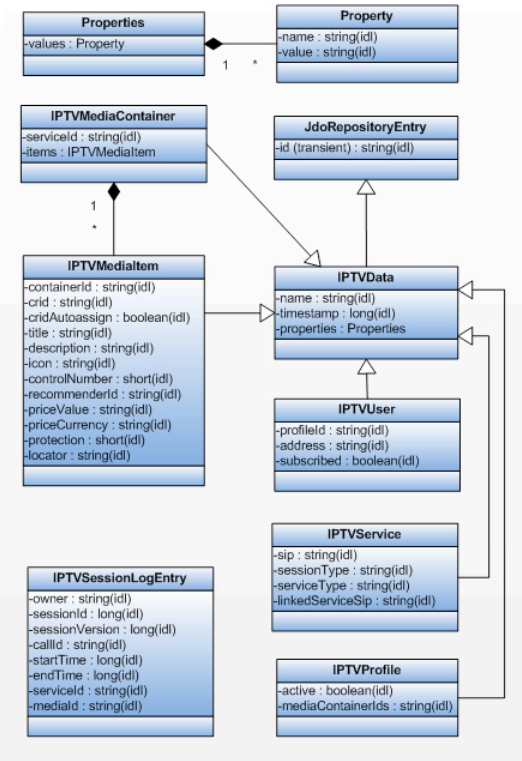

Figure 8: IPTV Database schema

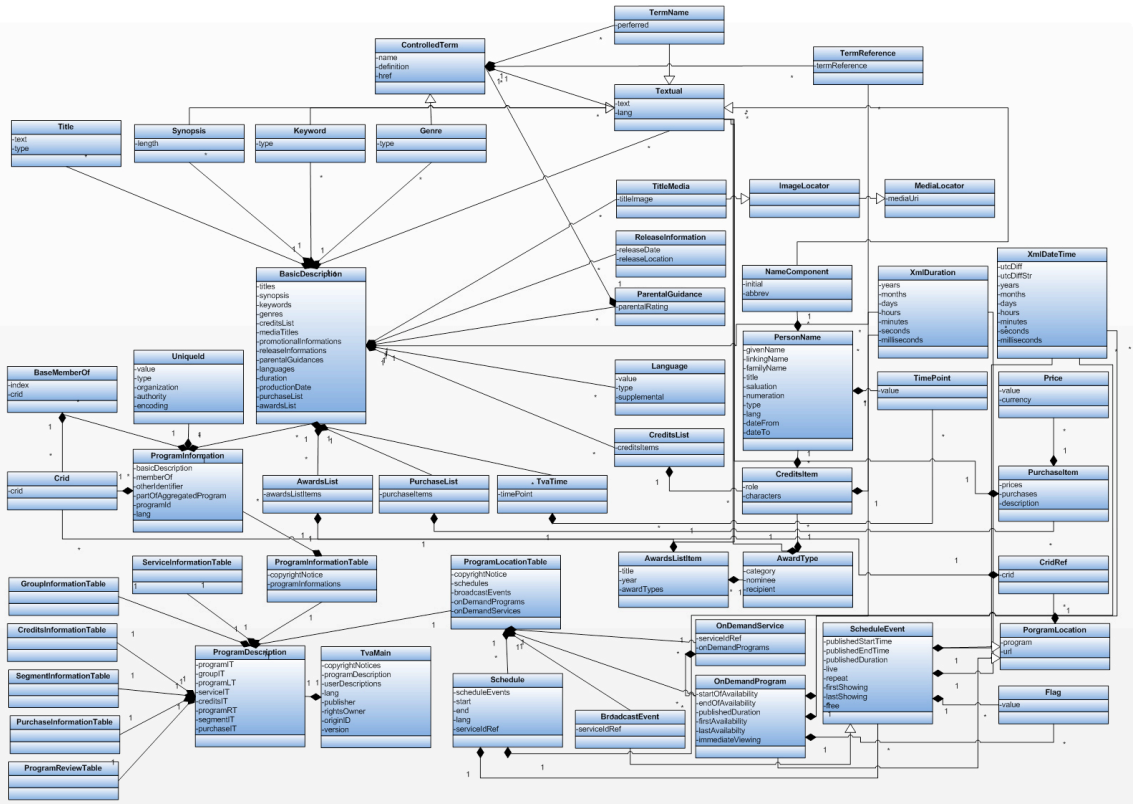

Figure 9: BCG Database schema

Benchmark Results and Analyses

Use Case 1: IPTV SC

The next pages show the benchmark results of internal (server-side) and external (network-side) INVITE and BYE requests. 10 SIP requests per second are sent to the database server . We also performed these test cases with a native memory repository (no persistence data) to analyze the approximate database overhead of each request. Following are the average results indicated in milliseconds.

Memory Benchmark Results

User	T_I	T_A	T_B	T_{NI}	T_{NB}	T_{NF}
1	1	1	1	11	5	57
100	3	1	1	11	7	60
10k	2	1	1	10	7	59

V/OD Benchmark Results

User	T_I	T_A	T_B	T_{NI}	T_{NB}	T_{NF}
1	6	4	5	13	9	61
100	8	5	7	16	12	66
10k	6	4	17	14	23	77

MySQL Benchmark Results

User	T_I	T_A	T_B	T_{NI}	T_{NB}	T_{NF}
1	13	5	7	21	11	63
100	12	9	10	18	15	68
10k	15	7	25	23	30	88

[1]

T_I = Average process time of an internal INVITE request (in milliseconds).

T_A = Average process time of an internal ACK request (in milliseconds).

T_B = Average process time of an internal BYE request (in milliseconds.

T_{NI} = Average process time of an external SIPINVITE request (in milliseconds).

T_{NB} = Average process time of an external SIPBYE request (in milliseconds).

T_{NF}= Overall average process time of each external test run (INV + ACK + BYE) (in milliseconds).

Following are graphs for the 10k user test case with internal and external setups.

[1] In the test network environment, this was the best applicable value to avoid retransmits and server locks.

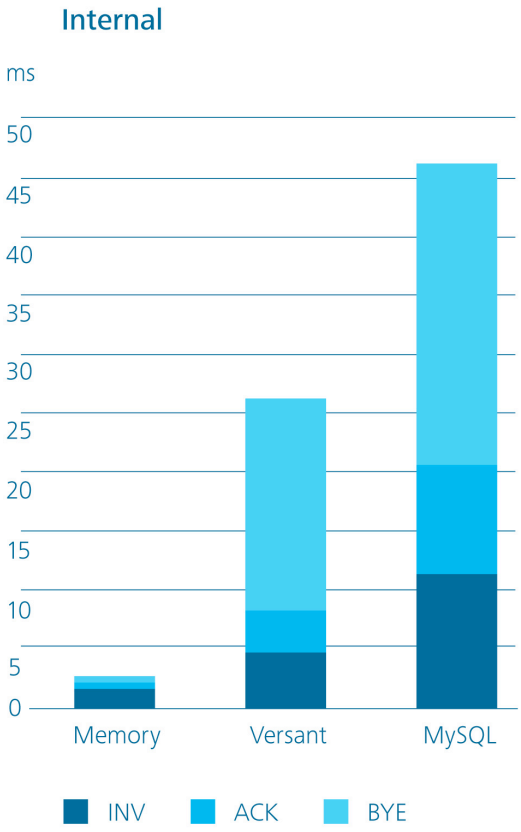

Figure 10: Internal process time results for 10k users

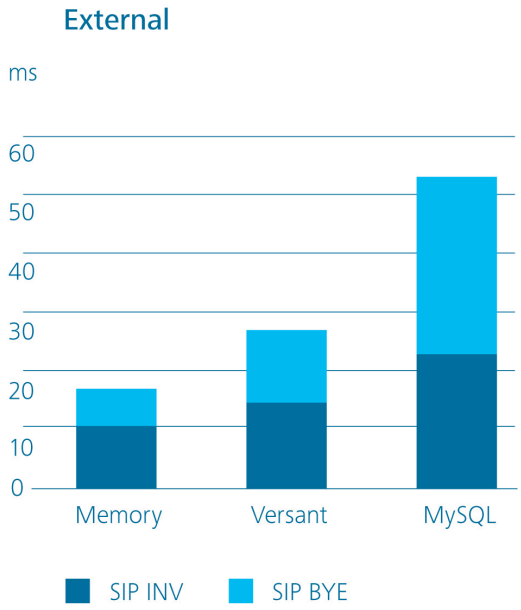

Figure 11: External process time results for 10k users

For the internal configuration both the application server and database server share the same machine. V/OD

performed more than 50% better compared to the MySQL. In the external configuration, V/OD performed approx, 25% better. The smaller advantages of the Versant Object Database over the network are based on the network IO bottleneck of the test environment, which could not handle the amount of incoming IO requests.

Use Case 2: Broadband Content Guide

The write and read times for BCG data are recorded and analyzed. The following table lists the processing time in seconds:

V/OD Benchmark Results

Channels	T_{WRITE}	T_{READ}
0	120,24	66,38
30	356,52	109,04
50	574,21	292,94

MySQL Benchmark Results

Channels	T_{WRITE}	T_{READ}
10	15,18	5,58
30	20,74	15,38
50	25,17	23,17

T_{WRITE} = Time for writing BCG data into the repository (in seconds).
T_{READ} = Time for reading BCG data from the repository (in seconds).

Figure 12 shows the processing time for filling/reading the repository with the converted BCG data, using Versant and MySQL repository implementations.

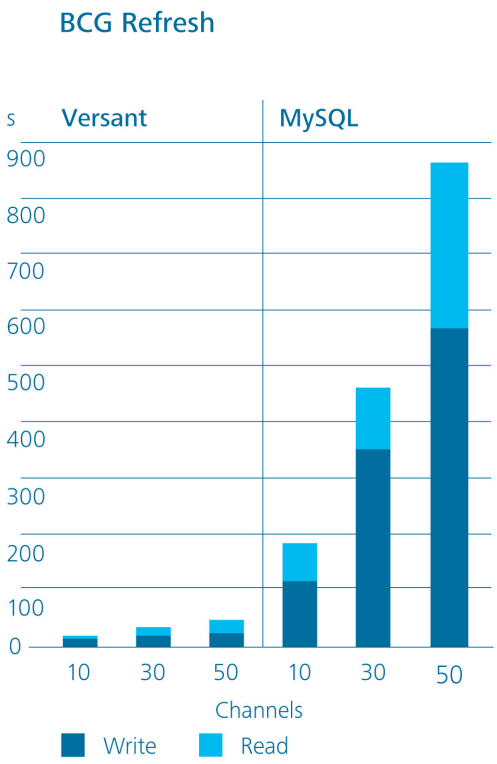

Figure 12: Process time results for BCG refresh

The results show a significant advantage for the V/OD managing complex data structures. The Versant implementation was over 8 times faster than MySQL. The MySQL database schema for BCG demonstrates the disadvantages of relational databases when managing a complex database model. The relational schema contains a large number of tables that must be joined together with references, when storing and retrieving BCG objects. This makes the MySQL database significantly slower.

JDO vs. Native MySQL

JDO is a high level application programming interface integrating transactional database capabilities into the Java language. JDO can work with both object oriented databases and relational databases, and since it is standardized, the application can run on different database systems without changing the application code.

JDO also encapsulates most if not all database connection logic. Therefore, the application programmer can concentrate entirely on developing the application logic and does not have to spend time writing mapping code from the application model into the database model.

JDO uses the Java application classes and methods to inherently create the database schema for the application developer. Creating a separate database schema is not required. We found the JDO API well-designed, easy to learn and fast to integrate with our application code. Due to these benefits, we would recommend using JDO in Java applications.

JDO Versant – MySQL DataNucleus compatibility

Comparing databases is never an easy task. Many specific requirements influence the decision making process, for example, the application specific data model and application specific data management tasks. Instead of using a standard database benchmark, such as TPC-C, we chose to develop a couple of benchmarks that are based on our existing IPTV Ecosystem data model and data management requirements, which allowed us to analyze results that are more relevant to the real world requirements found in such systems.

The objective of the first test case was to handle a demanding number of simultaneous read/write operations and queries with small data objects, typically found in an IPTV Session Control environment. V/OD performed more than 50% better compared to MySQL in a server side, 3-tier application server architecture. Our results for a traditional client/server architecture showed smaller advantages for the Versant Object Database, performing only approximately 25% better than MySQL, probably because of the network latency of the test environment.

The second test case managed larger sized BCG data in one large transaction. V/OD was more than 8 times faster compared to MySQL. In our analysis, the test case demonstrated V/OD's significant advantages when managing complex data structures.

Considering these results, we would recommend a V/OD database implementation where performance is mandatory and in particular when the application must manage complex data structures.

Furthermore, we discovered that the Java Data Objects persistence API provides a modern, easy to use programming interface to the Java programmer, which we can highly recommend to increase the productivity of the development team by alleviating the need to implement mapping code to the database.

Summary

The compatibility of the V/OD and the MySQL DataNucleus JDO implementations was flawless. The IPTV application can simply switch from one database system to the other by replacing the JDO property files containing the Persistence Manager Factory implementation definition and connection settings. It was not even necessary to re-enhance the data classes with the database specific enhancer (only the Versant enhancer was used). However developers must pay attention so that only JDO properties are used which both implementations support. Therefore, the usage of vendor-defined properties must be carefully planned, if they are required.